SAGE: Similarity-Based Cleaning of Poisoned Training Data from Verified Examples

Chaeeun Han*, Soodeh Atefi†, Yevgeniy Vorobeychik‡, and Aron Laszka*
*College of Information Sciences and Technology, Pennsylvania State University, University Park, PA, USA
†Department of Computer Science and Engineering, University of Louisville, Louisville, KY, USA
‡Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA
Email: {cfh5554, aql5923}@psu.edu, soodeh.atefi@louisville.edu, yvorobeychik@wustl.edu

Published in the Proceedings of the 6th ICDM Machine Learning for Cybersecurity Workshop (ICDM MLC 2026).

***Abstract*—As machine learning increasingly relies on public, untrusted data sources, data poisoning attacks, which inject malicious examples into training data to induce misclassification of a chosen target, pose a growing threat. Existing defenses either assume zero ground-truth information about which examples are poisoned, or they assume access to a large set of examples verified to be clean. Satisfying the latter assumption incurs significant cost since reliable verification can be very resource- or labor-intensive. This cost is particularly high for clean-label attacks, where poisoned examples are visually indistinguishable from clean data. Since requiring a large set of verified examples is impractical, we propose relying on a *small set of verified examples including both clean and poisoned ones*, i.e., each example verified either to be clean *or* poisoned through inspection by a forensic expert. The challenge is then to detect poisons based on a set of verified examples that is so small that most classification models would overfit. To address this challenge, we propose *Similarity-based Approach for Ground-truth-driven Exclusion* (SAGE), which trains a generic feature extractor on a separate dataset and then flags poisoned training examples using a non-parametric, similarity-weighted prediction based on the verified set. On standard benchmarks against seven clean-label attack methods, we demonstrate that having access to even a handful of verified poisoned examples provides a substantial advantage. We also find that the distribution of verified clean examples across classes matters more than the number of verified examples.**

***Index Terms*—data poisoning, clean-label attacks, data cleaning, similarity-based learning**

I. Introduction

Machine learning (ML) systems have become indispensable in a wide range of applications, from autonomous driving [1] to health care diagnostics [2]. The success of these systems hinges on the integrity and quality of their training data. However, as ML increasingly relies on external, untrusted data sources, it becomes more vulnerable to data poisoning attacks, in which adversaries introduce malicious examples into training data to cause misclassifications of specific test instances [3]–[6]. Such attacks can be practical: Carlini et al. [7] demonstrate that an adversary could have succeeded in achieving their misclassification goal by controlling a mere 0.01% of a web-scale dataset such as LAION-400M.

A number of cleaning approaches have been introduced to detect and remove poisoned examples from training data [3], [5], which vary in terms of their assumptions. Some cleaning approaches trust the class labels of the entire training set [8]; others assume nothing and rely on statistical outlier criteria, at the cost of lower precision [9], [10]. We focus on *base-set defenses* that assume access to a set of verified clean examples, which serve as a reference for the remaining data. The assumed size of this verified set varies between cleaning approaches: the frequency-based poison detector [11] and MW-Net [12] use 100 clean examples; while MNTD [13], Neural Cleanse [14] and I-BAU [15] use 1,000. De-Pois [16] assumes 20% of the training data to be verified, which corresponds to 10,000 verified images for CIFAR-10, a commonly used benchmark. Reducing the verified fraction degrades both its accuracy and the F1 score from roughly 0.9 to 0.7 as the fraction decreases to 0.5%. DUTI [17] likewise relies on a substantial number of verified clean examples.

However, verifying that an example is clean is expensive. Clean-label poisons are perturbed to remain visually indistinguishable from benign data, and professional inspectors struggle to identify over 86% of feature-perturbing poisons [18]. Verification therefore cannot rely on rapid visual screening; it requires deliberate per-example scrutiny, whose cost grows linearly with the size of the verified set. For this reason, gathering the hundreds or thousands of verified clean examples that existing base-set defenses require is often impractical.

Further, the assumption that verification provides only clean examples may also be impractical. An inspection process that is meticulous enough to reliably verify examples as clean may also confirm some examples to be poisoned, and these carry more information than what clean examples carry alone. A verified clean set characterizes only how clean data are distributed, so a defense built on it must treat every deviation from this distribution as suspect. A set of verified poisons instead marks regions of the feature space that the attack targets, enabling defense to focus on these specific regions. We therefore study a setting in which the defender has access to both kinds of verified examples, and we show that relying on both enables effective detection based on verified sets that are much smaller than those assumed by existing base-set defenses.

Given both kinds of verified examples, the detection of remaining poisons can itself be posed as an ML problem: a binary classification task separating poisoned examples from clean examples, trained on the set of verified examples and applied to the rest of the training data. For the kind of

unstructured data that we consider throughout this work, such as images, training a classifier based on a few dozen verified examples is very challenging. Training a model directly on so few examples immediately overfits the model. Fitting a classifier on features output by a fixed, pretrained feature extractor only shifts the difficulty: clean-label poisons differ from benign data by small perturbations, so any representation exposing them must retain fine-grained detail and is therefore high-dimensional, leaving the decision boundary under-determined, creating a curse of dimensionality for the detection problem. Both failures share a cause: each fits parameters to the verified set, which is far too small to determine them. This motivates an approach that fits no parameters to that set at all.

In this work, we propose **SAGE** (**S**imilarity-based **A**pproach for **G**round-truth-driven **E**xclusion), which addresses these challenges. SAGE trains a feature extractor on a separate dataset, then classifies each candidate as poisoned or clean by a non-parametric, similarity-weighted prediction over the verified set. Because no parameters are fit to that set, its size affects only how finely the prediction can discriminate, not the risk of overfitting. This shifts the technical question from *how to classify* to *how to represent*. An extractor trained with a standard classification objective is optimized to make classes linearly separable, not to place similar examples near each other, which is what SAGE relies on at detection time. We therefore train it with an objective that mirrors detection. Each batch is split into a reference subset and a query subset, predictions for the query subset are formed as a similarity-weighted combination of reference labels, and the resulting cross-entropy is back propagated through the extractor. Detection then applies the same rule with the verified examples as the reference set.

We evaluate SAGE on CIFAR-10 [19] and Tiny ImageNet [20] against three clean-label triggerless attacks (feature collision [21], bullseye polytope [22], and gradient matching [23]) and against four clean-label backdoor attacks (Narcissus [24] and three variants of Wicked Oddities [25]). On CIFAR-10, SAGE finds at least 480 of the 500 poisoned examples in every triggerless setting, limiting attack success rate (ASR) to at most 5%, while discarding less than 2% of the training data. Against backdoor attacks, SAGE is the only defense that effectively suppresses attacks without discarding too much training data: the only baseline method that attains significantly lower ASR does so by discarding so much training data that the accuracy of the trained application model falls below 0.50.

The remainder of this paper is organized as follows. Section II reviews related work on data poisoning and defenses, and explains how our assumptions differ. Section III introduces the threat model and our objective. Section IV presents our proposed method, SAGE. Section V describes our experimental setup, and Section VI reports results. Section VII discusses the limitations of our approach and provides concluding remarks.

II. Related Work

A. Data Poisoning Attacks

Data poisoning attacks compromise machine learning (ML) systems by injecting malicious examples into training datasets, disrupting model performance [3], [6], [26]. Unlike evasion attacks, which manipulate test-phase inputs without corrupting the model itself, poisoning attacks occur during the training phase [27]. This fundamental difference allows poisoning attacks to have long-lasting effects on the model's behavior.

Attacks are commonly categorized along two axes. The first distinguishes untargeted attacks, which degrade overall performance [6], from targeted attacks, which cause specific test instances to be misclassified [28]. The second distinguishes label-flipping attacks, which alter labels [29], from clean-label attacks, which perturb features while leaving the original labels intact [21]. Clean-label attacks are the stealthier of the two because each poisoned example remains consistent with its label. The attack survives inspections that would immediately expose a flipped label.

Their practicality has been established at scale. Carlini et al. [7] show that an adversary could have controlled 0.01% of a web-scale dataset such as LAION-400M or COYO-700M by purchasing expired domains that the dataset indexes, thereby enabling data poisoning attacks. We consider three representative clean-label targeted attacks: feature collision [21], which optimizes poisons to collide with a target in feature space; bullseye polytope [22], which surrounds the target with a convex hull of poisons; and gradient matching [23], which aligns poison gradients with those of the adversarial objective.

B. Defense Strategies against Data Poisoning

Defenses that filter suspected poisons from the training set [30], [31] differ mainly in what they assume the defender already knows. We organize them accordingly.

a) Label-dependent defenses: These trust the class labels of the entire training set. Deep k-NN [8] flags a point as poisoned when its own label disagrees with the plurality label among its k nearest neighbors in feature space. The assumption suits clean-label attacks, where labels are correct by construction, and Deep k-NN is a strong baseline in exactly that regime; it provides no advantages when labels themselves are manipulated.

b) Ground-truth-agnostic defenses: These assume nothing, identifying poisons using statistical, outlier-based, or clustering criteria: spectral signatures [9], activation clustering [32], and gradient-based core-set selection [10]. Freedom from assumptions is costly. Meta-Sift [18] evaluates on five criteria on CIFAR-10 and finds that none of them isolates clean data with sufficient precision, with several performing worse than random selection under at least one attack. In practice, these methods tend toward over-filtering, discarding large volumes of clean data to catch the poisons [33].

c) Base-set defenses: These assume access to a set of verified clean examples, against which the remaining data is compared. Assumed sizes vary widely: the frequency-based

poison detector [11] and MW-Net [12] use 100 examples, MNTD [13], Neural Cleanse [14] and I-BAU [15] use 1,000, and De-Pois [16] adopts 20% of the training data by default—10,000 images on CIFAR-10—reporting that accuracy and F1 fall from roughly 0.9 to 0.7 as this fraction shrinks toward 0.5%. DUTI [17] likewise relies on a set of trusted examples.

The difficulty with this third group is not the size of the assumed set, but its assumed purity. Zeng et al. [18] show that across five representative base-set defenses, performance collapses once less than 1% of the base set is poisoned. For the frequency-based detector, a single poisoned image among 100 drives performance below the random baseline. The verification step that would prevent this is also unreliable for precisely the attacks we study. Across 16 attacks inspected by 48 professional annotators, human inspectors missed over 86% of poisons that perturb features while leaving labels intact, close to chance. A large presumed-clean set is thus not simply expensive to obtain. At the sizes these defenses assume, contamination is likely, and contamination is what breaks these defenses.

SAGE differs from all three. It does not trust the labels of the untrusted data, does not operate without ground truth, and does not assume a pure base set. Instead, it assumes a small verified set in which examples are labeled as clean *or* poisoned, and uses both. Relative to label-dependent defenses, this trades a large volume of assumed-correct class labels for a much smaller number of integrity judgments, which survive label manipulation. Relative to ground-truth-agnostic defenses, it supplies the anchor those methods lack, at the cost of an assumption they avoid. Relative to base-set defenses, it removes the purity requirement, since verified poisons are labeled as such rather than presumed absent. Verification of this kind is also realistic. Data forensics, incident response, and expert or crowdsourced inspection [34] all produce examples of both kinds, yet existing defenses retain only half of what they yield.

III. Problem Setting and Threat Model

a) Setting: A defender trains a classifier f_θ on a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ with $y_i \in \{1, \dots, K\}$. An attacker corrupts a small subset by replacing $m \ll n$ *base* examples x_{b_j} with *poison* examples x_{p_j}. Depending on the attack, the poison is either an additive perturbation $x_{p_j} = x_{b_j} + \delta_j$ with $\|\delta_j\|_\infty \le \epsilon$, or a localized patch overwriting a small region of x_{b_j}. In both cases the modification is small enough that the poison is not obviously anomalous to a human inspector. Labels are left unchanged, $y_{p_j} = y_{b_j}$.

b) Attacker: The attacker aims for a model trained on D' to misclassify a chosen set of test inputs T into an attacker-chosen label y_b, while behavior on all other inputs and hence overall test accuracy remains largely unaffected. In *targeted* attacks, $T = \{x_t\}$ is a single test example with a true label $y_t \neq y_b$, and no modification is made at test time. In *backdoor* attacks, T is the set of inputs carrying the attacker's trigger patch, so the attacker controls a whole region of input space rather than a single point.

Because labels are not altered, no label inconsistency is obviously visible for a human to catch. The poisoned model appears normal under aggregate accuracy monitoring in both cases: in the targeted setting because only one input is affected, and in the backdoor setting because the trigger is absent from clean test data.

c) Defender: The defender observes only D', and knows neither m, nor which examples were poisoned, nor x_t, y_b, or ϵ, nor which attack was used. The same procedure is applied regardless of the underlying attack.

Beyond the entire possibly poisoned dataset D', the defender has access to a small *ground-truth set* $G = G_{\text{clean}} \cup G_{\text{poison}} \subset D'$ of examples independently verified as clean or poisoned, with $|G| \ll n$. We use $g_j \in \{0, 1\}$ for the verification label of example $j \in G$, with $g_j = 1$ indicating a poison, and $q = |G_{\text{poison}}|$ for the number of verified poisons. We assume G_{clean} is class-balanced, containing c verified clean samples from each of the K classes.

d) Goal: The defender produces a cleaned dataset $\hat{D} \subseteq D'$, obtaining parameters $\theta(\hat{D})$. Given an unknown set of poisoned samples $P \subset D'$, the defense should satisfy three requirements at the same time. First, the attack should fail on the attacker's targets, so that $f_{\theta(\hat{D})}(x_t) \neq y_b$. Second, accuracy on cleaned data should be close to what training on the original clean D would have achieved. Lastly, the number of examples discarded, $|D' \setminus \hat{D}|$, should be small. None of x_t, y_b, or P is observed.

These three requirements are in tension. Discarding most of D' can satisfy the first one but fails the others, while discarding nothing may satisfy the last two but fails the first one. The quantity that distinguishes defenses is therefore not attack success alone, but also the amount of data discarded to achieve it, and we report both throughout Section VI.

IV. Proposed Approach: SAGE

SAGE (Similarity-based Approach for Ground-truth-driven Exclusion) identifies poisoned training examples by comparing them, in a learned feature space, to a small set of verified poisoned and clean examples. It proceeds in two stages. A feature extractor ϕ is first trained on CIFAR-100 so that images sharing a label are placed close together (Section IV-B). At defense time, the frozen ϕ is reused to interpolate the verified set's poison/clean designations over that same similarity structure (Section IV-C).

A. Feature Extractor

Rather than adopting a pretrained vision model, we train a feature extractor from scratch on a separate dataset. The verified set is far too small to train on directly, so the representation must come from elsewhere; and a pretrained model is optimized for a linear classifier on top of its features, whereas SAGE compares examples by cosine similarity. Training our own extractor also lets us set the output dimensionality directly, which we tune for similarity-based detection (Appendix A).

The extractor maps an image $\mathbf{x}$ to a feature vector $\mathbf{z} = \phi(\mathbf{x}) \in \mathbb{R}^d$ by adding a two-layer head to a ResNet-18 backbone:

$$\mathbf{z} = \phi(\mathbf{x}) = \mathbf{W}_2\, \sigma\big(\mathrm{BN}(\mathbf{W}_1\, \mathbf{r}(\mathbf{x}) + \mathbf{b}_1)\big) + \mathbf{b}_2, \qquad (1)$$

where $\mathbf{r}(\mathbf{x}) \in \mathbb{R}^{512}$ is the ResNet-18's pooled feature, σ is a sigmoid, and BN is batch normalization. The head maps $\mathbf{r}(\mathbf{x})$ to a hidden width h and then projects to the output dimension d, the space in which we calculate similarity. Alongside ϕ, we also learn a scale parameter $\tau > 0$, which controls how sharply the prediction (Equation (3)) concentrates on the most similar references.

B. Training the Feature Extractor

Each training batch is split into a *reference* set R, whose labels are treated as known, and a *query* set Q, whose labels must be predicted from R alone by similarity. A query sample is classified correctly only when same-label reference samples receive high similarity weight. Since that weight comes from distance in feature space, minimizing the loss forces ϕ to place same-label samples close together. At detection time the same extractor and the same prediction rule are reused, with the verified set in place of R and verification labels in place of class labels.

The extractor is trained on CIFAR-100 [19], disjoint from CIFAR-10 and Tiny ImageNet, the datasets attacked in our experiments (Section V); this separation ensures the defense is not biased by the datasets under attack. We expect the resulting space to transfer reasonably to both targets: all three are standard image benchmarks with broadly similar visual statistics, and training against 100 diverse classes rather than a handful discourages ϕ from specializing to a few categories—the same reasoning behind why models pretrained on large, diverse label sets transfer to unrelated downstream tasks.

a) Training Step: Each training step consists of three sub-steps. *1. Similarity calculation:* Every image in $R \cup Q$ passes through ϕ. For a query image i and a reference image j, we compute the cosine similarity between their features,

$$\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\| \, \|\mathbf{z}_j\|}. \tag{2}$$

2. Prediction: The predicted label for query sample i is a similarity-weighted average of the reference labels, with weights given by a softmax over similarity at scale τ:

$$\hat{\mathbf{y}}_i = \frac{\sum_{j \in R} \exp\big(\tau \cdot \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)\big)\, \mathbf{y}_j}{\sum_{j \in R} \exp\big(\tau \cdot \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)\big)}, \tag{3}$$

where $\mathbf{y}_j$ is the known label of reference sample j—at this stage, its one-hot CIFAR-100 label—so that $\hat{\mathbf{y}}_i$ is a distribution over the C classes. Higher values of τ narrow the prediction toward the nearest references, while lower values draw on the reference set more evenly.

3. Loss calculation and backpropagation: Let c_i denote the true class index of query sample i. Whereas Equation (3) uses only the *reference* labels, the loss compares $\hat{\mathbf{y}}_i$ against the *query* sample's own label, giving the cross-entropy

$$\mathcal{L} = -\frac{1}{|Q|} \sum_{i \in Q} \log \hat{y}_i^{(c_i)}. \tag{4}$$

$\mathcal{L}$ is back propagated through Equation (3) and Equation (1), updating the weights of ϕ and the scale τ jointly. Gradients flow through the features of both the reference and query samples since both are produced by ϕ.

b) Construction of the Split: Sampling of R and Q must be class-stratified rather than uniform. Under a uniform split, most query samples have no reference sample of their own class in the same step, and Equation (3) cannot assign weight to a class it has not seen; the achievable loss is then floored and training makes no progress. We therefore draw k subsets per step, each containing s samples from every one of the C classes, and rotate which subset serves as R, while the remaining $k-1$ form Q, so that every sample serves in both roles across the k inner iterations. Drawing a fresh stratified sample each step keeps the cost per step fixed while letting the reference/query pairing vary across training. The values of k, s, and the optimization settings are given in Section V.

Because τ is calibrated against $|R|$ during training and then reused unchanged at detection time, where the reference set is the much smaller verified set G, a large mismatch between $|R|$ and $|G|$ leaves τ tuned for a setting the detector never encounters. We therefore treat $|R|$ as a hyperparameter and report results for a range of $|R|$ in Appendix B.

Algorithm 1 SAGE feature extractor training

Require: Dataset D with C classes, extractor ϕ

Initialize $\tau \leftarrow 1$
for *epoch* $\leftarrow 1$ to N **do**
  for each step **do**
    Draw k class-stratified subsets $S_1, \ldots, S_k$ from D, each with s images per class
    for $r \leftarrow 1$ to k **do**
      $R \leftarrow S_r$; $\quad Q \leftarrow \bigcup_{l \neq r} S_l$
      $\mathbf{z} \leftarrow \phi(\mathbf{x})$ for all $\mathbf{x} \in R \cup Q$
      Compute $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)$ for $i \in Q$, $j \in R$ ▷ Eq. (2)
      Compute $\hat{\mathbf{y}}_i$ for $i \in Q$ ▷ Eq. (3)
      Compute $\mathcal{L}$ ▷ Eq. (4)
      Update ϕ and τ by gradient descent on $\mathcal{L}$
        ▷ gradients flow through both R and Q
    end for
  end for
end for

C. Applying SAGE as a Poison Detector

At detection time we reuse Equations (2) and (3) with g_j in place of $\mathbf{y_j}$, with two differences from training. First, ϕ and τ are frozen: no training or fine-tuning is performed on the target dataset. Second, the reference set is the verified set, $R = G$, and reference labels are no longer one-hot class vectors $\mathbf{y}_j$ but scalar poison indicators $g_j \in \{0, 1\}$, with $g_j = 1$ for $j \in G_{\text{poison}}$ and $g_j = 0$ for $j \in G_{\text{clean}}$. So, each reference image retains its own class label as well as its poison status.

Every remaining training example takes the query role. For a candidate sample $\mathbf{x}_i$, Equation (3) returns

$$\hat{g}_i = \frac{\sum_{j \in G} \exp\big(\tau \cdot \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)\big)\, g_j}{\sum_{j \in G} \exp\big(\tau \cdot \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)\big)} \in [0, 1], \tag{5}$$

the similarity-weighted fraction of verified poisons among the reference set—an estimated poison probability. No loss is computed at this stage, since the candidate's true status is what is being inferred, and no parameters are updated. We remove $\mathbf{x}_i$ from the training set when $\hat{g}_i \geq 0.5$. Examples in G_{clean} are retained regardless, having already been verified.

V. Experimental Setup

A. Datasets

We evaluate SAGE on CIFAR-10 and Tiny ImageNet, two standard benchmarks in the data-poisoning literature. CIFAR-10 contains 50,000 training images across 10 classes. Tiny ImageNet contains 100,000 training images across 200 classes. Neither is used to train the feature extractor, which is trained on CIFAR-100 (Section V-B).

Experiments run on a server with a 64-core 2.4 GHz AMD EPYC CPU and an NVIDIA RTX A5000 GPU (24 GB). Training the feature extractor takes roughly 30 minutes. Once trained, applying SAGE to a 50,000-image training set takes 4 minutes. The extractor is trained once and reused across every attack, dataset, and verified-set configuration reported below, so the marginal cost of each additional experiment is the 4-minute detection pass.

B. Feature Extractor Training

The extractor of Equation (1) uses a ResNet-18 backbone trained from scratch on CIFAR-100 [19], with a two-layer head of hidden width $h = 512$ and output dimension $d = 256$. Optimization uses SGD (learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, cosine-annealed) for 200 epochs, with random crops and horizontal flips. The scale τ is initialized to 1 and optimized jointly with the network weights.

Reference/query splits are drawn as described in Section IV-B: at each step we sample $k = 4$ class-stratified subsets of $s = 2$ images per class, giving subsets of 200 images and a step size of $k \times s \times C = 800$ images, with roughly 62 steps per epoch. The reference set at training time therefore has $|R| = 200$.

C. Verified Set

Our default verified set on CIFAR-10 contains $c = 2$ clean examples from each of the $K = 10$ classes and $q = 2$ confirmed poisons drawn from the base class, giving $|G| = 22$. Poisons and clean examples are drawn from the poisoned training set itself, so $G \subset D'$; no external data is assumed. Section VI-D varies both the total size of G and its per-class composition.

D. Victim Model Training

After filtering, we train a victim model on the cleaned dataset. On CIFAR-10 we use white-box transfer learning with ResNet-18 for Feature Collision and Bullseye Polytope (40 epochs, SGD, learning rate 0.1, batch size 128), and train ResNet-18 from scratch for Gradient Matching (200 epochs, same optimizer settings). On Tiny ImageNet we train VGG-16 from scratch (200 epochs, learning rate 0.1, batch size 64).

For the trigger-based attacks in Section VI-B we train ResNet-18 on CIFAR-10 from scratch with a single fixed recipe: SGD, learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, cosine-annealed, batch size 128, 100 epochs, random crop and flip. Holding this recipe constant across attacks ensures that differences in defended ASR and accuracy reflect the attack and defense rather than the training procedure; each attack's own configuration (trigger design, poison rate, target-class restriction) still follows its published setting.

E. Poisoning Attacks

a) Triggerless attacks: We follow the benchmark of Schwarzschild et al. [35] for the three triggerless attacks, adopting its hyperparameters, optimizers, and evaluation protocol. Results are reported over 20 trials with distinct base–target pairs. We evaluate $\epsilon \in \{8, 16, 22\}$, where 8 and 16 are common triggerless poisoning budgets and 22 is included as an additional stronger-perturbation setting. Each setting generates 500 poisons for CIFAR-10 and 250 for Tiny ImageNet.

Feature Collision [21] optimizes poisons to collide with the target in feature space, so that a decision boundary accommodating the poisons captures the target. The attacker knows the pretrained feature extractor the victim will fine-tune.

Bullseye Polytope [22] instead surrounds the target with a convex hull of poisons in feature space, placing the target near its center, which relaxes the collision constraint and transfers more reliably. The attacker again knows the pretrained extractor.

Gradient Matching [23] aligns the gradient of the training loss on the poisons with that of the adversarial objective, and is effective against models trained from scratch. Here the attacker knows only the victim's architecture and training procedure, not the resulting weights.

b) Backdoor attacks: For backdoor attacks, we use each author's released implementation and default hyperparameters. For backdoor attacks, ASR is measured over the triggered test set rather than a single target sample; we also report clean accuracy. Both are clean-label: the trigger is applied only to images already belonging to the target class, and no label is altered.

Narcissus [24] optimizes a global trigger without any knowledge of the victim's training set. The attacker trains a surrogate model on a public out-of-distribution dataset, appends the target class to it, and then searches for a perturbation that drives inputs strongly toward that class.

Wicked Oddities [25] keeps the trigger fixed and selects which images of the target-class to poison, motivated by the observation that samples contribute unequally to attack success. As the selection strategy is trigger-agnostic, we experiment with three instantiations: Wicked-BadNets, Wicked-Blended, and Wicked-SIG.

F. Baseline Defenses

Deep k-NN [8] flags a point whose label disagrees with the plurality label among its k nearest neighbors in feature space. It is the most directly comparable baseline for triggerless attacks. It requires no verified set, only the class labels of the

training data, which under a triggerless attack are correct by construction. Following the default setting, we set k to the number of examples per class.

EPIC [10] clusters examples in gradient space and drops points isolated from the rest during training, requiring no ground-truth information.

Meta-Sift [18] selects a clean subset of a user-specified size from a poisoned dataset. Following its default 1,000 sample selection budget, we use the complementary ranking as a filter and discard the 1,000 lowest-ranked images (leaving 49,000 for CIFAR-10 and 99,000 for Tiny ImageNet).

Confusion Training [33] decouples benign correlations by training on poisoned data alongside randomly mislabeled clean samples, so that the model fits poisoned samples while failing to fit clean ones, exposing the poisons.

Activation Clustering [32] clusters the final-layer activations of each class into two groups and flags the smaller one as poisoned, on the assumption that poisoned samples form a distinct cluster within their label.

SPECTRE [36] estimates robust mean and covariance of each class's representations and whitens them, amplifying the directions in which poisoned samples deviate from the clean distribution. The samples are then ranked by a QUE score along these directions, and the highest-scoring ones are removed.

FLARE [37] aggregates abnormal activations across all hidden layers rather than the final one, arguing that poison/benign separability varies by depth. It selects an adaptive subspace, splits the dataset into two clusters, and flags the more stable cluster as poisoned.

G. Metrics

We report attack success rate (ASR) and clean test accuracy. For the three targeted attacks, ASR is the fraction of trials in which the single target image x_t is classified as the attacker's chosen base label. Test accuracy is measured on the standard test split after training on the cleaned dataset. We additionally report the fraction of the training set removed and the false-positive rate among removed examples (i.e., removed samples being benign), since a defense that removes a large share of clean data can suppress ASR while degrading the model it was meant to protect.

VI. Numerical Results

A. Triggerless Attacks

Table I reports the post-defense attack success rate (ASR) and clean test accuracy on CIFAR-10 and Tiny ImageNet. Without a defense, ASR varies substantially across attacks. Bullseye Polytope succeeds in 95–100% of trials, and Gradient Matching succeeds in 75–100%. In contrast, Feature Collision succeeds in only 15–35% of trials. Results for Feature Collision should therefore be interpreted relative to its lower undefended ASR.

On CIFAR-10, SAGE achieves the lowest ASR in every setting, including ties, reducing ASR to 0% for Feature Collision and Bullseye Polytope at all perturbation budgets. For Gradient Matching, it limits ASR to 0–5%. EPIC also achieves 0% ASR for Feature Collision at all three budgets. For Bullseye Polytope at $\epsilon = 16$, SAGE achieves 0% ASR, tied with Deep k-NN, compared with 15% for EPIC, 35% for Meta-Sift, and 70% for Confusion Training.

The accuracy results are more mixed. Under transfer learning, EPIC and Deep k-NN achieve higher clean accuracy than SAGE, with ranges of 0.9455–0.9466 and 0.9389–0.9481, respectively, compared with 0.9258–0.9428 for SAGE. The ordering reverses under from-scratch training for Gradient Matching, where SAGE retains 0.8980–0.9033 against 0.8787–0.8806 for EPIC, though Confusion Training is best at 0.9040–0.9122. Relative to the undefended models, SAGE costs 0.3–5.1 percentage points, with the largest reductions under from-scratch training.

a) Tiny ImageNet: We evaluate Bullseye Polytope on Tiny ImageNet using VGG-16. SAGE reduces ASR from 30% to 10% and slightly improves accuracy from 0.6153 to 0.6169. It is the only defense that avoids an accuracy loss in this setting. Deep k-NN reduces ASR to 20%, but its accuracy falls to 0.5584. EPIC leaves ASR unchanged at 30% and reduces accuracy to 0.5569.

B. Backdoor Attacks

Table II reports results for the four clean-label backdoor attacks on CIFAR-10. Each poisons 10% of an attacker-selected target class, corresponding to 1% of the full training set. We measure ASR over the 9,000 test images outside the target class after applying the trigger.

SAGE reduces ASR for all four attacks, reaching 0.1310 on Narcissus and 0.2720 on Wicked-BadNets. Wicked-Blended has the highest residual ASR under SAGE at 0.4620, followed by Wicked-SIG at 0.3400. Clean accuracy decreases by 2.40 percentage points on Narcissus and by at most 0.50 percentage points on the Wicked variants. On Narcissus, Activation Clustering [32], SPECTRE [36], and Confusion Training achieve lower ASR than SAGE, but also lower clean accuracy. On all three Wicked variants, SAGE achieves lower ASR than these defenses while maintaining clean accuracy above 0.94.

FLARE [37] achieves the lowest ASR on the three Wicked variants, but its clean accuracy ranges from 0.4586 to 0.5507 across the four attacks, compared with 0.9281–0.9494 for SAGE. These results highlight the trade-off between attack suppression and clean accuracy: although SAGE does not achieve the lowest ASR overall, it substantially reduces ASR across all four attacks while retaining higher clean accuracy than FLARE.

C. Precision of Removal

Table III reports the total number of removed samples and the number of true poisons among them. These results reveal substantial differences in removal precision.

On CIFAR-10, SAGE recovers more than 480 of the 500 poisons while removing 769.2–995.1 images in total, or 1.5–2.0% of the training set, with 35.4–50.8% false positives. The baselines trade off differently. Meta-Sift removes 1,000 images in every setting but recovers at most 105.1 poisons, a recall up to 21%. EPIC removes 9,189.5–9,290 images under transfer

TABLE I

POST-DEFENSE ATTACK SUCCESS RATE (ASR) AND CLEAN TEST ACCURACY (ACC.) FOR CLEAN-LABEL TRIGGERLESS POISONING ATTACKS ON CIFAR-10 AND TINY IMAGENET. RESULTS ARE AVERAGED OVER 20 POISONED DATASETS FOR EACH ATTACK AND PERTURBATION BUDGET ϵ. ASR IS THE FRACTION OF SUCCESSFUL ATTACKS ACROSS THE 20 POISONED DATASETS, WHILE ACC. IS REPORTED AS MEAN $\pm$ STANDARD DEVIATION. FOR EACH ATTACK, THE BEST RESULT AMONG THE DEFENSES IS SHOWN IN BOLD, AND THE SECOND-BEST RESULT IS UNDERLINED.

Attack		No Defense		SAGE (Ours)		Deep k-NN		Meta-Sift		EPIC		Confusion Training	
Size	Method	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)
CIFAR-10													
ϵ = 8	FC	0.9460±0.0008	15%	0.9258±0.0017	**0%**	0.9389±0.0034	**0%**	0.9233±0.0003	15%	0.9465±0.0008	**0%**	**0.9474±0.0042**	5%
	BP	0.9457±0.0007	100%	0.9366±0.0012	**0%**	**0.9471±0.0002**	20%	0.9155±0.0002	25%	0.9466±0.0008	10%	0.9416±0.0046	45%
	GM	0.9487±0.0006	75%	0.8980±0.0025	**0%**	0.8973±0.0123	10%	0.8907±0.0009	20%	0.8787±0.0048	10%	**0.9040±0.0065**	10%
ϵ = 16	FC	0.9456±0.0008	25%	0.9392±0.0024	**0%**	0.9426±0.0032	10%	0.9255±0.0003	25%	**0.9465±0.0006**	**0%**	0.9451±0.0032	5%
	BP	0.9458±0.0009	100%	0.9375±0.0031	**0%**	0.9389±0.0015	**0%**	0.9242±0.0002	35%	**0.9455±0.0008**	15%	0.9334±0.0033	70%
	GM	0.9486±0.0008	95%	0.8986±0.0021	**5%**	0.9007±0.0032	10%	0.8952±0.0009	40%	0.8794±0.0035	**5%**	**0.9051±0.0047**	15%
ϵ = 22	FC	0.9455±0.0008	35%	0.9428±0.0048	**0%**	**0.9481±0.0020**	10%	0.9258±0.0003	35%	0.9465±0.0009	**0%**	0.9417±0.0034	20%
	BP	0.9458±0.0007	95%	0.9428±0.0041	**0%**	0.9477±0.0032	10%	0.9258±0.0002	40%	0.9465±0.0008	**0%**	**0.9500±0.0029**	25%
	GM	0.9491±0.0009	100%	0.9033±0.0029	**5%**	0.8985±0.0039	**5%**	0.8837±0.0011	60%	0.8806±0.0023	**5%**	**0.9122±0.0064**	15%
Tiny ImageNet													
ϵ = 8	BP	0.6153±0.0047	30%	**0.6169±0.006**	**10%**	0.5584±0.0051	20%	0.5869±0.0021	50%	0.5569±0.0047	30%	0.6025±0.0057	25%

TABLE II

POST-DEFENSE CLEAN TEST ACCURACY (ACC.) AND ATTACK SUCCESS RATE (ASR) FOR FOUR CLEAN-LABEL BACKDOOR ATTACKS ON CIFAR-10. RESULTS ARE MEAN $\pm$ STD OVER 10 RUNS. FOR EACH ATTACK, THE BEST RESULT AMONG THE DEFENSES IS SHOWN IN BOLD, AND THE SECOND-BEST RESULT IS UNDERLINED. NOTE THAT FLARE'S LOW ASR COMES AT THE COST OF EXCESSIVELY LOW CLEAN ACCURACY.

Attack	No Defense		SAGE (Ours)		Activation Clustering		SPECTRE		Confusion Training		FLARE	
	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)	Acc. (↑)	ASR (↓)
CIFAR-10												
Narcissus	0.9521±0.0004	0.9120±0.0010	**0.9281±0.0030**	0.1310±0.0017	0.9023±0.0020	**0.0431±0.0025**	0.9045±0.0006	0.1235±0.0290	0.9149±0.0010	0.0987±0.0061	0.5364±0.0294	0.0452±0.0011
Wicked-BadNets	0.9467±0.0006	0.8970±0.0035	**0.9494±0.0022**	0.2720±0.0048	0.9469±0.0006	0.8915±0.0067	0.9425±0.0055	0.8959±0.0174	0.9475±0.0049	0.8065±0.0074	0.5507±0.0392	**0.0699±0.0469**
Wicked-Blended	0.9471±0.0002	0.9383± 0.0007	**0.9467±0.0007**	0.4620±0.0030	0.9447±0.0029	0.9381±0.0014	0.9428±0.0047	0.5608±0.0758	0.9408±0.0042	0.5675±0.0054	0.4586±0.0040	**0.1023±0.0762**
Wicked-SIG	0.9451±0.0007	0.9372±0.0018	0.9401±0.0022	0.3400±0.0157	0.9407±0.0056	0.9382±0.0262	0.9400±0.0054	0.7613±0.0475	**0.9452±0.0041**	0.7612±0.0038	0.4941±0.0354	**0.0717±0.0206**

learning and about 19,405.1–20,174.2, around 38.8–40.3% of the dataset, from scratch, in the latter case recovering fewer than 100 poisons. Deep k-NN removes the fewest images, 205.3–406.4, but 46.1–90.7% of those are false positives.

For Gradient Matching at $\epsilon = 16$, SAGE and EPIC both achieve an ASR of 5%, but remove an average of 896.9 and 20,174.2 images, respectively. Thus, the same ASR can conceal a large difference in the amount of training data discarded, motivating the reporting of removal counts alongside ASR.

The pattern differs on Tiny ImageNet. SAGE removes 187 images, including 126 of the 250 poisons. This result corresponds to 50.4% recall and a 32.6% of removed examples being benign, the best removal precision among the evaluated defenses. Despite removing only about half of the poisons, SAGE reduces ASR from 30% to 10%. This result suggests that the attack depends on retaining a sufficiently large fraction of its poisons.

D. Composition of the Verified Set

We next vary the composition of the verified set while holding all other components fixed. The feature extractor is trained once and reused across configurations. Each configuration therefore requires only an additional detection pass.

Figure 1 compares two verified-set sampling strategies as the number of verified poisons increases. With class-balanced clean examples, SAGE recovers most poisons using only two verified poisons for BP-8, FC-8, and GM-8. With unstratified clean examples, approximately five verified poisons are needed to achieve comparable recall.

TABLE III

EXAMPLES IDENTIFIED AS POISONED BY EACH DEFENSE. TP DENOTES THE NUMBER OF TRUE POISONS IDENTIFIED, AND TOTAL DENOTES THE TOTAL NUMBER OF EXAMPLES REMOVED. THE DATASETS CONTAIN 500 POISONS ON CIFAR-10 AND 250 POISONS ON TINY IMAGENET. RESULTS ARE MEAN $\pm$ STD OVER 20 RUNS.

Attack		SAGE		Deep k-NN		Meta-Sift		EPIC	
Size	Method	TP (↑)	Total	TP (↑)	Total	TP (↑)	Total	TP (↑)	Total
CIFAR-10									
$\epsilon = 8$	FC	489.6 ± 23.4	995.1 ± 34.1	34.0 ± 33.4	220.8 ± 31.0	37.1 ± 2.4	1,000.0 ± 0.0	388.1 ± 54.1	9,191.8 ± 220.9
	BP	500.0 ± 0.0	847.2 ± 14.5	64.5 ± 53.0	251.8 ± 51.2	69.5 ± 18.4	1,000.0 ± 0.0	448.2 ± 50.4	9,242.5 ± 148.1
	GM	482.1 ± 14.3	844.3 ± 13.7	19.0 ± 14.2	205.3 ± 13.4	24.3 ± 4.0	1,000.0 ± 0.0	58.2 ± 30.2	19,405.1 ± 446.5
$\epsilon = 16$	FC	496.9 ± 2.5	769.2 ± 13.8	85.4 ± 69.2	273.1 ± 68.0	42.3 ± 3.4	1,000.0 ± 0.0	454.0 ± 35.7	9,189.5 ± 205.7
	BP	495.5 ± 2.2	831.6 ± 14.9	175.6 ± 113.3	363.2 ± 111.5	84.4 ± 49.6	1,000.0 ± 0.0	480.7 ± 69.0	9,237.0 ± 188.7
	GM	480.4 ± 13.5	896.9 ± 23.7	39.0 ± 34.8	225.3 ± 35.2	50.6 ± 28.4	1,000.0 ± 0.0	59.5 ± 35.2	20,174.2 ± 648.9
$\epsilon = 22$	FC	496.2 ± 1.1	830.7 ± 14.4	115.2 ± 92.5	303.2 ± 91.1	29.7 ± 5.9	1,000.0 ± 0.0	374.2 ± 28.9	9,196.3 ± 215.4
	BP	499.8 ± 0.5	858.2 ± 8.9	219.2 ± 123.9	406.4 ± 122.1	105.1 ± 59.8	1,000.0 ± 0.0	426.6 ± 75.0	9,290.0 ± 177.8
	GM	496.2 ± 1.5	827.1 ± 13.8	56.4 ± 30.5	242.4 ± 29.9	59.2 ± 35.9	1,000.0 ± 0.0	57.4 ± 30.4	20,002.4 ± 945.1
Tiny ImageNet									
$\epsilon = 8$	BP	126.0 ± 38.5	187.0 ± 50.4	77.7 ± 11.8	220.4 ± 18.4	3.3 ± 3.4	1,000.0 ± 0.0	87.4 ± 5.3	40,036.0 ± 945.1

The difference is more pronounced for false positives. Class-balanced sampling removes approximately 180–220 clean images. Unstratified sampling removes approximately 1,500–2,200 clean images, even when it achieves lower recall. These results show that balanced coverage of the clean classes is important for controlling false positives.

VII. DISCUSSION AND CONCLUSION

A. Adaptive Attacks

We consider how an adaptive attacker might target SAGE directly. An adaptive attacker could attempt to lower the similarity $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)$ between each poison and the verified poisons in G_{poison}. Alternatively, the attacker could increase its similarity to the examples in G_{clean}. Either strategy could reduce

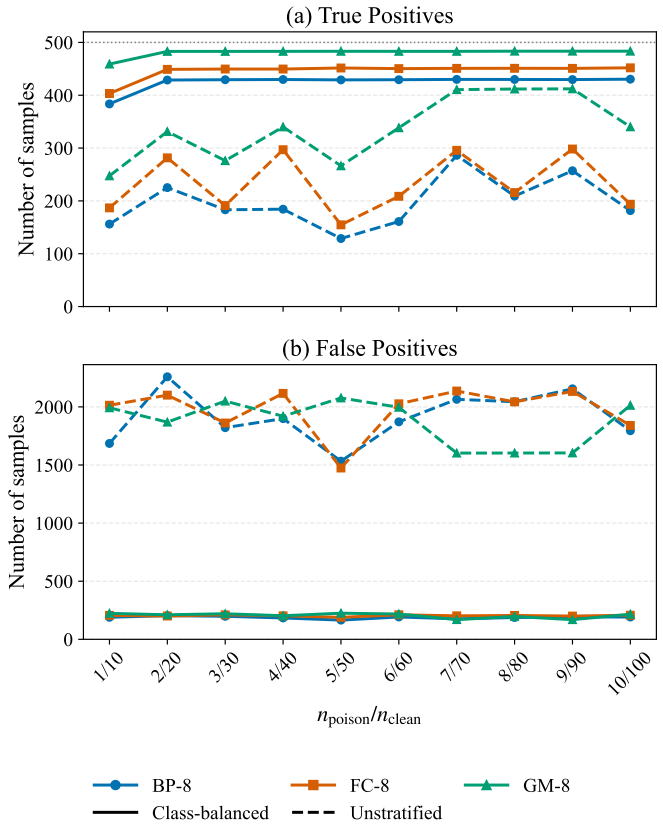

Fig. 1. Effect of verified-set composition on poison recall and false positives. The upper panel reports the number of poisoned images identified, and the lower panel reports the number of clean images removed. Solid lines denote class-balanced sampling of verified clean examples. Dashed lines denote unstratified random sampling.

the poison probability returned by Equation (3). However, two aspects of our setting complicate such attacks.

First, ϕ is trained on CIFAR-100, disjoint from the attacked datasets. It is also not exposed to the poisoning. An attacker without access to the weights of ϕ cannot directly compute or differentiate through the similarity function used by SAGE.

Second, even a white-box attacker with access to ϕ does not know which examples will form G. The verified set is built independently of the attack, typically by inspecting a small sample of training data after poisoning has occurred. The attacker can therefore target only the distribution of clean and poisoned embeddings, but not the specific reference examples used for detection. This separation is part of our threat model, not a general robustness guarantee; we make no claim against an attacker who knows G when constructing the poisons.

A more fundamental line of attack targets the assumption that verified poisons are representative of unverified ones. SAGE relies on poisoned examples occupying related regions of the feature space. An attacker could challenge this by increasing diversity among the poisons, sacrificing the effectiveness of each individual poison in exchange for lower similarity across poisons. Unlike the strategies above, this one is not blocked by the attacker's uncertainty about ϕ or G, since it targets the central premise of SAGE rather than the specific reference examples used at detection time.

Evaluating these adaptive strategies is an important direction for future work. One approach is to add an evasion term to the Bullseye Polytope or Gradient Matching objective, penalizing proximity to a verified set. Another is to construct diversity-regularized variants of these attacks and measure the resulting trade-off between attack effectiveness and detectability.

B. Limitations

a) Assumption of correct verified set: SAGE has a weaker form of the assumption made by the base-set defenses: we do not require a large verified set to be pure, but we do assume that examples identified as clean are indeed clean. A poison incorrectly included in G_{clean} would provide an erroneous reference for the similarity-weighted prediction. The assumption is easier to satisfy at our scale, since certifying a few dozen examples is more tractable than certifying the hundreds or thousands used elsewhere, but our experiments do not evaluate contamination of G_{clean} at any scale. Measuring how performance degrades under such contamination is an important robustness experiment left for future work.

b) Cross-dataset reuse of the feature extractor: We train ϕ on CIFAR-100 and reuse it without fine-tuning on CIFAR-10 and Tiny ImageNet. This design avoids retraining for each dataset, but the learned embedding reflects the class structure of CIFAR-100 rather than that of the target dataset. Our experiments show that the embedding separates clean and poisoned examples in the evaluated settings, though the method does not guarantee separation elsewhere. Performance may decline when the target domain differs from natural images, such as in medical imaging, and extending the method to non-visual modalities would require separate evaluation.

C. Conclusion

This work proposes SAGE, a defense against clean-label poisoning that assumes a small set of examples verified as clean or poisoned, rather than a large base set presumed clean, and classifies the remaining data by similarity to them in a learned feature space. On CIFAR-10, SAGE holds attack success to at most 5% against Feature Collision, Bullseye Polytope, and Gradient Matching while recovering at least 480 of the 500 poisons, on average. On Tiny ImageNet, SAGE reduces Bullseye Polytope ASR from 30% to 10% while recovering 126 of 250 poisons. It also suppresses four clean-label backdoor attacks without the accuracy loss incurred by the one baseline reaching comparable success rates. What separates it from existing defenses is how little it discards: against Gradient Matching on CIFAR-10, SAGE removes fewer than 900 images on average at every tested perturbation budget, whereas EPIC removes more than 19,000. Knowing even a handful of poisoned examples is thus disproportionately valuable, and verification effort is better spent identifying examples of both kinds than assembling a larger set assumed to be clean.

ACKNOWLEDGMENTS

This work was partially supported by the Army Research Office (W911NF-25-1-0059) and Office of Naval Research (N000142412663).

REFERENCES

[1] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4439–4449, 2020.

[2] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and S. Rab, "Significance of machine learning in healthcare: Features, pillars and applications," *International Journal of Intelligent Networks*, vol. 3, pp. 58–73, 2022.

[3] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.

[4] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–36, 2022.

[5] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo, and F. Roli, "Wild patterns reloaded: A survey of machine learning security against training data poisoning," *ACM Computing Surveys*, vol. 55, no. 13, pp. 1–39, 2023.

[6] J. Fan, Q. Yan, M. Li, G. Qu, and Y. Xiao, "A survey on data poisoning attacks and defenses," in *7th IEEE International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2022, pp. 48–55.

[7] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *45th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2024, pp. 407–425.

[8] N. Peri, N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-NN defense against clean-label data poisoning attacks," in *16th European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 55–70.

[9] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[10] Y. Yang, T. Y. Liu, and B. Mirzasoleiman, "Not all poisons are created equal: Robust training against data poisoning," in *39th International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 25 154–25 165.

[11] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2021, pp. 16 453–16 461.

[12] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[13] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI trojans using meta neural analysis," in *42nd IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021, pp. 103–120.

[14] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *40th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 707–723.

[15] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia, "Adversarial unlearning of backdoors via implicit hypergradient," in *9th International Conference on Learning Representations (ICLR)*, 2021.

[16] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3412–3425, 2021.

[17] X. Zhang, X. Zhu, and S. Wright, "Training set debugging using trusted items," in *32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, vol. 32, 2018.

[18] Y. Zeng, M. Pan, H. Jahagirdar, M. Jin, L. Lyu, and R. Jia, "Meta-Sift: How to sift out a clean subset in the presence of data poisoning?" in *32nd USENIX Security Symposium (USENIX Security '23)*, 2023, pp. 1667–1684.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, April 2009.

[20] Y. Le and X. S. Yang, "Tiny ImageNet visual recognition challenge," 2015.

[21] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[22] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," in *6th IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 159–178.

[23] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," in *9th International Conference on Learning Representations (ICLR)*, 2021.

[24] Y. Zeng, M. Pan, H. A. Just, L. Lyu, M. Qiu, and R. Jia, "Narcissus: A practical clean-label backdoor attack with limited information," in *30th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023, pp. 771–785.

[25] H. Q. Nguyen, H. Nguyen, A. Ta, T. Nguyen-Tang, K.-S. Wong, T. T. Hoang, and K. Doan, "Wicked oddities: Selectively poisoning for effective clean-label backdoor attacks," in *13th International Conference on Learning Representations (ICLR)*, vol. 2025, 2025, pp. 10 999–11 021.

[26] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, 2022.

[27] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *2013 European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD'13)*. Springer, 2013, pp. 387–402.

[28] J. Guo and C. Liu, "Practical poisoning attacks on neural networks," in *16th European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 142–158.

[29] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *20th European Conference on Artificial Intelligence (ECAI 2012)*. IOS Press, 2012, pp. 870–875.

[30] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[31] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *Machine Learning*, vol. 111, pp. 1–47, 2022.

[32] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[33] X. Qi, T. Xie, J. T. Wang, T. Wu, S. Mahloujifar, and P. Mittal, "Towards a proactive ML approach for detecting backdoor poison samples," in *32nd USENIX Security Symposium (USENIX Security '23)*, 2023, pp. 1685–1702.

[34] F. Tahmasebian, L. Xiong, M. Sotoodeh, and V. Sunderam, "Crowdsourcing under data poisoning attacks: A comparative study," in *34th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec 2020)*. Springer, 2020, pp. 310–332.

[35] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, "Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks," in *38th International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 9389–9398.

[36] J. Hayase, W. Kong, R. Somani, and S. Oh, "SPECTRE: Defending against backdoor attacks using robust statistics," in *38th International Conference on Machine Learning (ICML)*. PMLR, July 2021, pp. 4129–4139.

[37] L. Hou, W. Luo, Z. Hua, S. Chen, L. Y. Zhang, and Y. Li, "FLARE: Toward universal dataset purification against backdoor attacks," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 6459–6473, 2025.

APPENDIX

The feature extractor ϕ is trained once on CIFAR-100 and reused for all detection experiments, so its two design choices—the output dimension d and the training reference-set size $|R|$—are fixed before any poisoned data is seen. We select both by measuring cross-entropy loss and accuracy on the clean CIFAR-100 test set under the similarity-weighted prediction rule in Equation (3), the same rule used at detection time.

A. Feature Extractor Output Dimension

Table IV evaluates four output dimensions for the feature extractor. We report cross-entropy loss and accuracy on the clean CIFAR-100 test set using the reference-query prediction rule in Equation (3). Performance is not monotonic in the output dimension. The dimension $d = 256$ achieves the lowest loss of 2.0417 and the highest accuracy of 0.7328. Increasing the dimension to 512 slightly degrades both metrics. This result suggests that the projection provides a useful bottleneck rather than simply limiting model capacity. We therefore use $d = 256$ in all experiments. The learned scale τ ranges from 13.03 to 13.56 across the evaluated dimensions and is largest at $d = 256$.

TABLE IV
EFFECT OF THE FEATURE-EXTRACTOR OUTPUT DIMENSION d. CROSS-ENTROPY LOSS AND ACCURACY ARE MEASURED ON THE CLEAN CIFAR-100 TEST SET USING THE SIMILARITY-WEIGHTED PREDICTION RULE IN EQUATION (3).

Dimension	**Scale τ**	**Loss**	**Accuracy**
64	13.5630	2.3252	0.7020
128	13.0431	2.2413	0.7146
256	13.0456	**2.0417**	**0.7328**
512	13.0370	2.1955	0.7242

B. Reference Set Size

The scale τ is learned during training and reused during detection. The size of the training reference set R is therefore an important design choice. Table V shows that accuracy is highest at $|R| = 200$ and decreases for both smaller and larger reference sets.

The result for $|R| = 22$ is particularly informative. With 100 classes, such a small reference set covers only about one fifth of the classes. Most queries therefore lack a same-class reference. In response, the learned value of τ approaches zero, and the prediction rule becomes close to a uniform average. These results indicate that class coverage, rather than reference-set size alone, is important. At $|R| = 200$, the extractor receives an average of two references per class. We therefore use $|R| = 200$.

TABLE V
EFFECT OF THE REFERENCE-SET SIZE $|R|$ USED DURING FEATURE-EXTRACTOR TRAINING. ACCURACY IS MEASURED ON THE CLEAN CIFAR-100 TEST SET USING THE SIMILARITY-WEIGHTED PREDICTION RULE IN EQUATION (3).

$\lvert R\rvert$	**Scale τ**	**Accuracy**
22	0.0000	0.5166
100	1.2895	0.6386
200	3.5260	**0.7350**
500	2.0171	0.6867